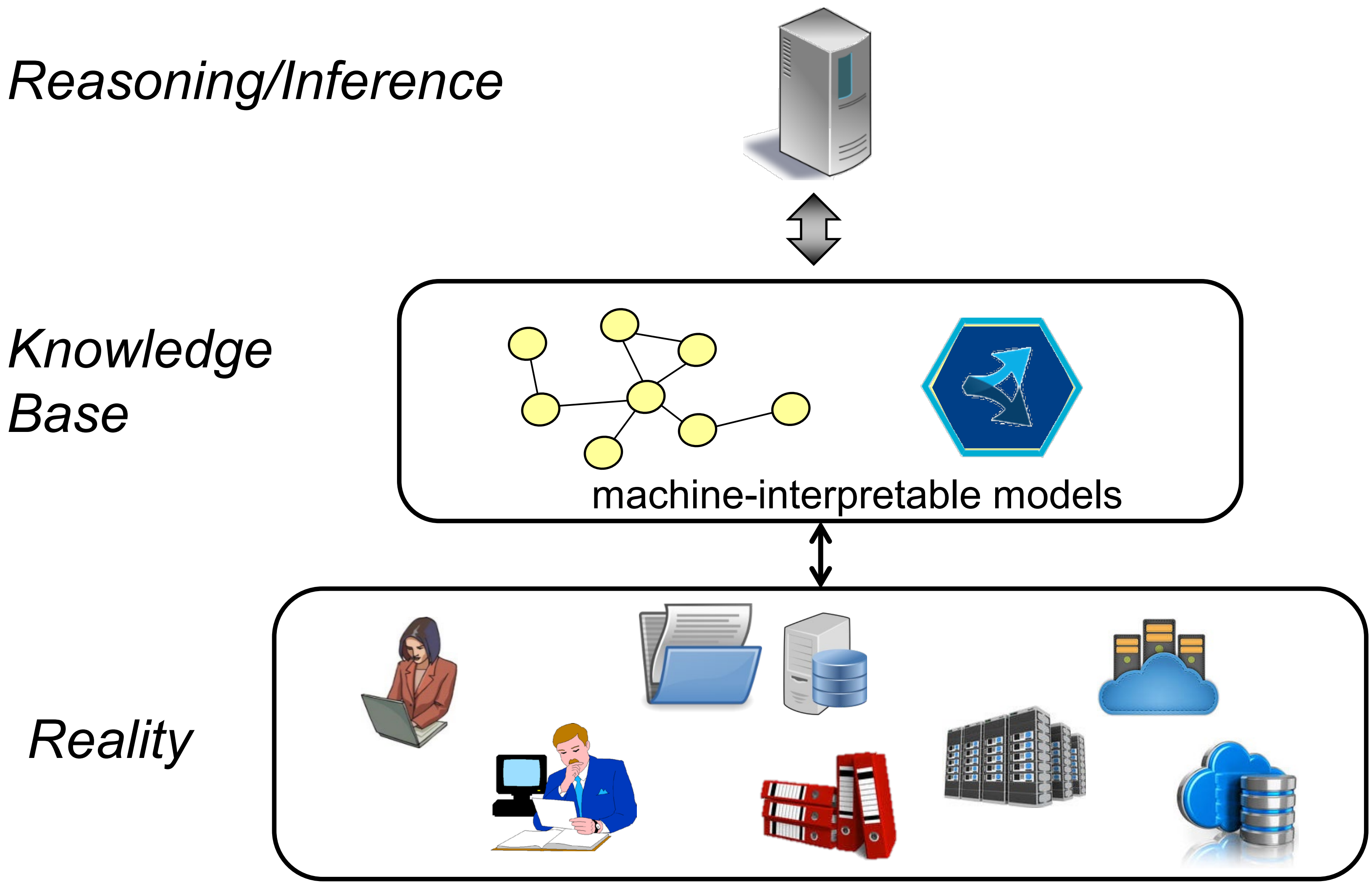
Ontology Engineering

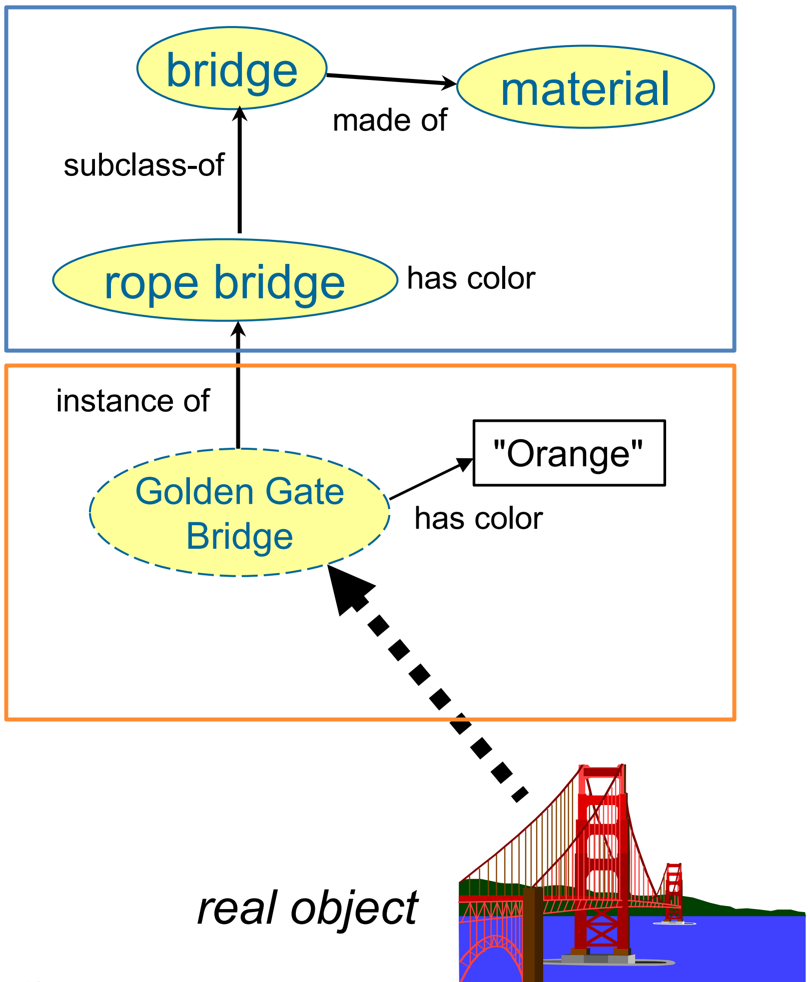
Knut Hinkelmann

Knowledge-Representation and Reasoning



Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

An Ontology – very informal



An ontology is a formal explicit description of concepts in a domain of discourse

- An ***ontology*** consists of
  - Concepts (Classes),
  - Relationships (Object Properties) between concepts
  - Attributes (Data Properties) of concepts
  - Constraints that hold between/for the concepts,
- An ontology together with a set of individual instances constitutes a ***knowledge base***

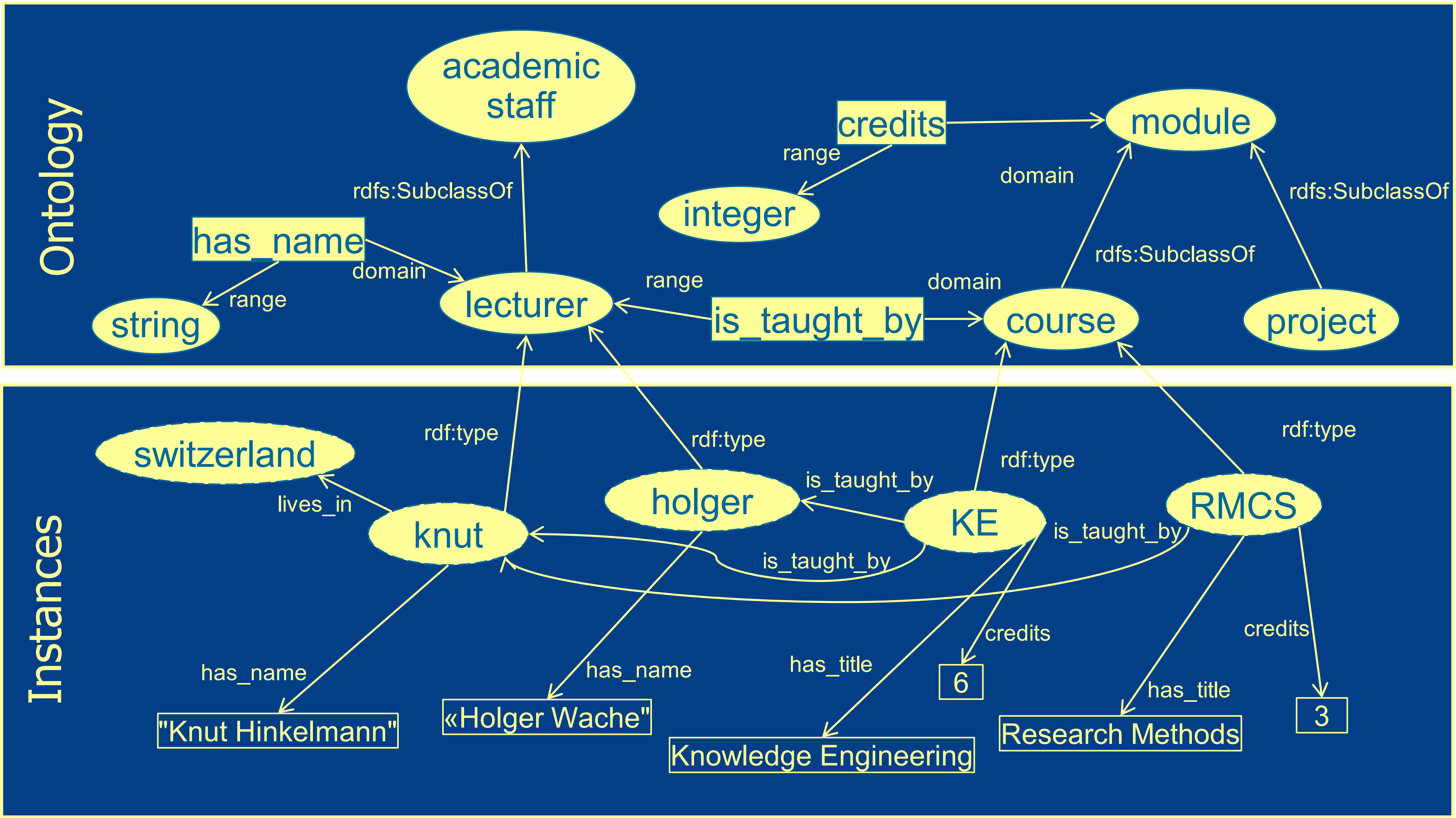
https://people.cs.uct.ac.za/~mkeet/OEbook/slides/L1IntroOE19.pptx ontology engineering

is

knowledge engineering

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Example of an Ontology

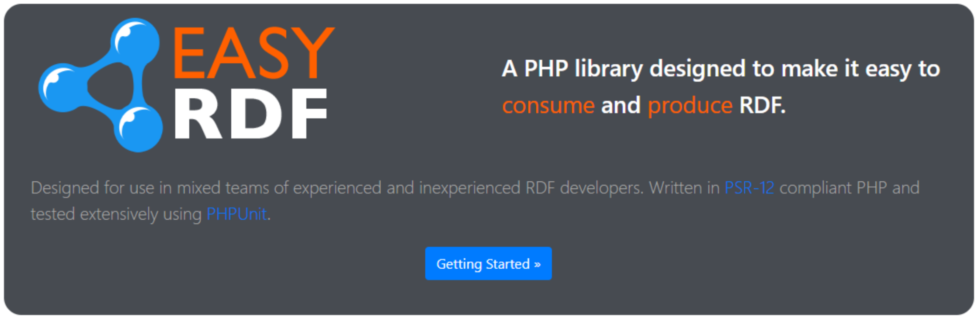


Ontology Representation Formalisms

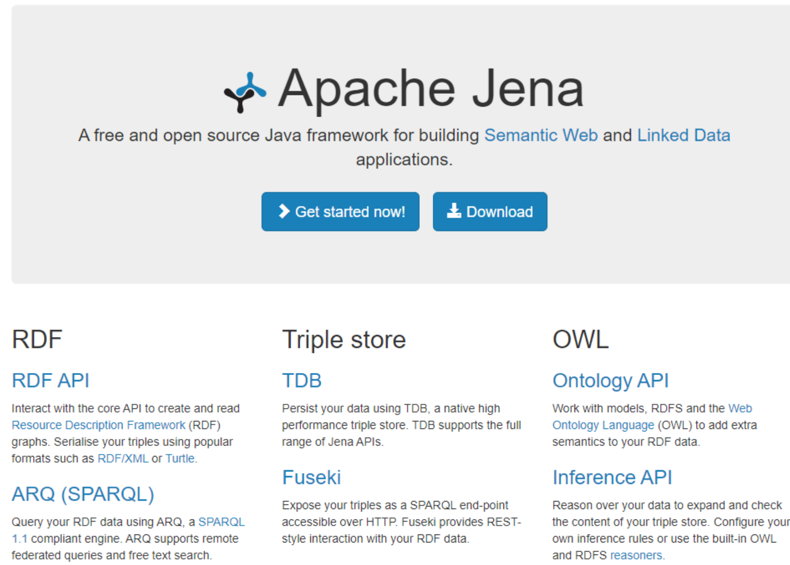
- Representations of Ontologies
  - ***RDF(S)*** ⇐ Our focus
  - OWL
  - Neo4J
  - …

Tools: Examples of Programming Libraries

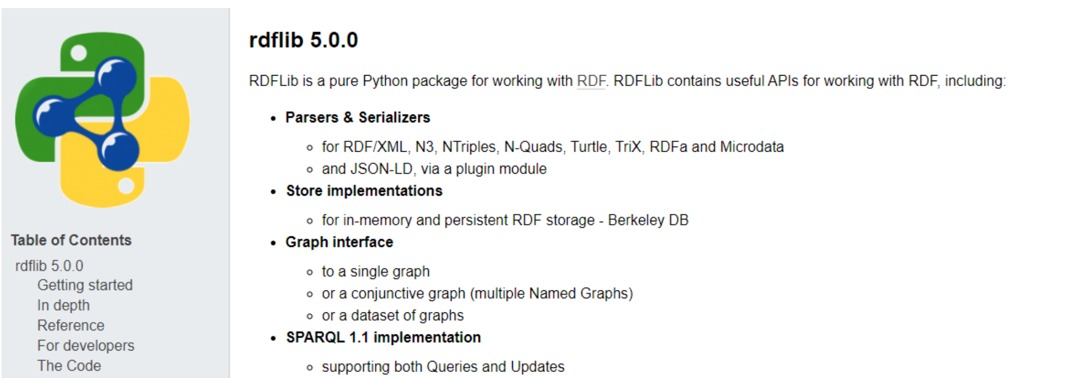
EasyRDF for PHP: https://www.easyrdf.org/



Apache Jena for Java: https://jena.apache.org/



RDFLib for Python: https://rdflib.readthedocs.io/en/stable/



Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Tool: Ontology Engineering

https://protege.stanford.edu/

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

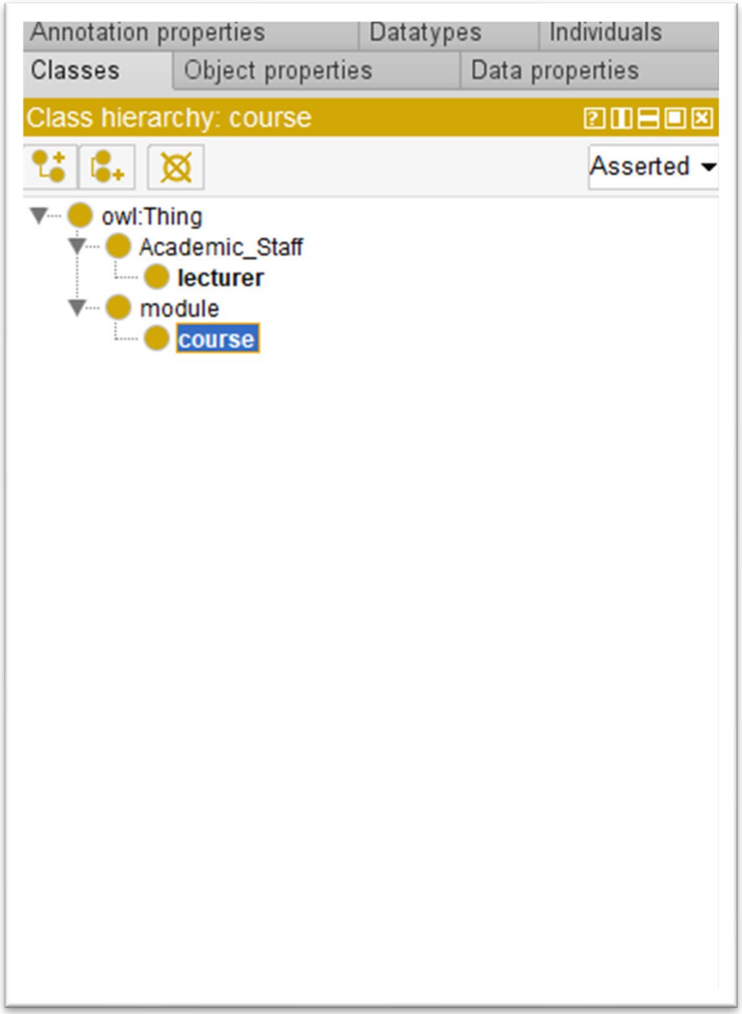
Creating an Ontology

- Defining classes in the ontology
- Arranging the classes in a taxonomic (subclass-superclass) hierarchy
- Defining properties and describing allowed values for the properties
- Creating instances and filling the values for properties

Define Classes and Class Hierarchy

- There are several approaches
  - Top-down: Start with the most general concept, and work your way down
  - Bottom-up: Start with the most specific, and work your way up
  - Combination

```
:Academic_Staff rdf:type owl:Class .
:lecturer rdf:type owl:Class ;
          rdfs:subClassOf :Academic_Staff .
:module rdf:type owl:Class .
:course rdf:type owl:Class ;
        rdfs:subClassOf :module .
```

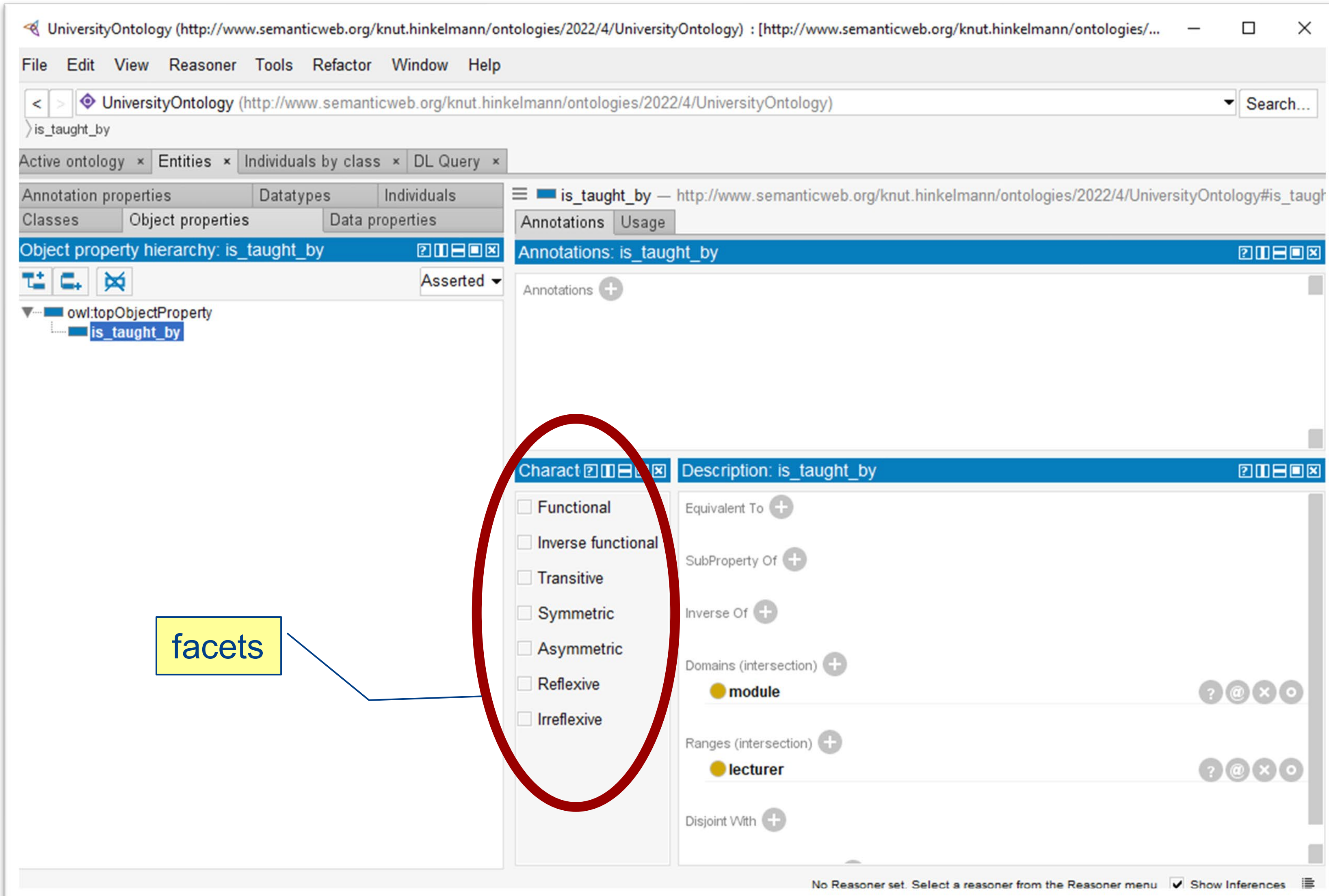


Define Properties of Classes

- Describe the internal structure of concepts
  - Data Properties: Attributes
    - Range are data types like String, Integer, …
  - Object Properties: Relations to other concepts
    - Range are Classes
- Desribe facets: Characteristics of Properties
- Inheritance to Subclasses

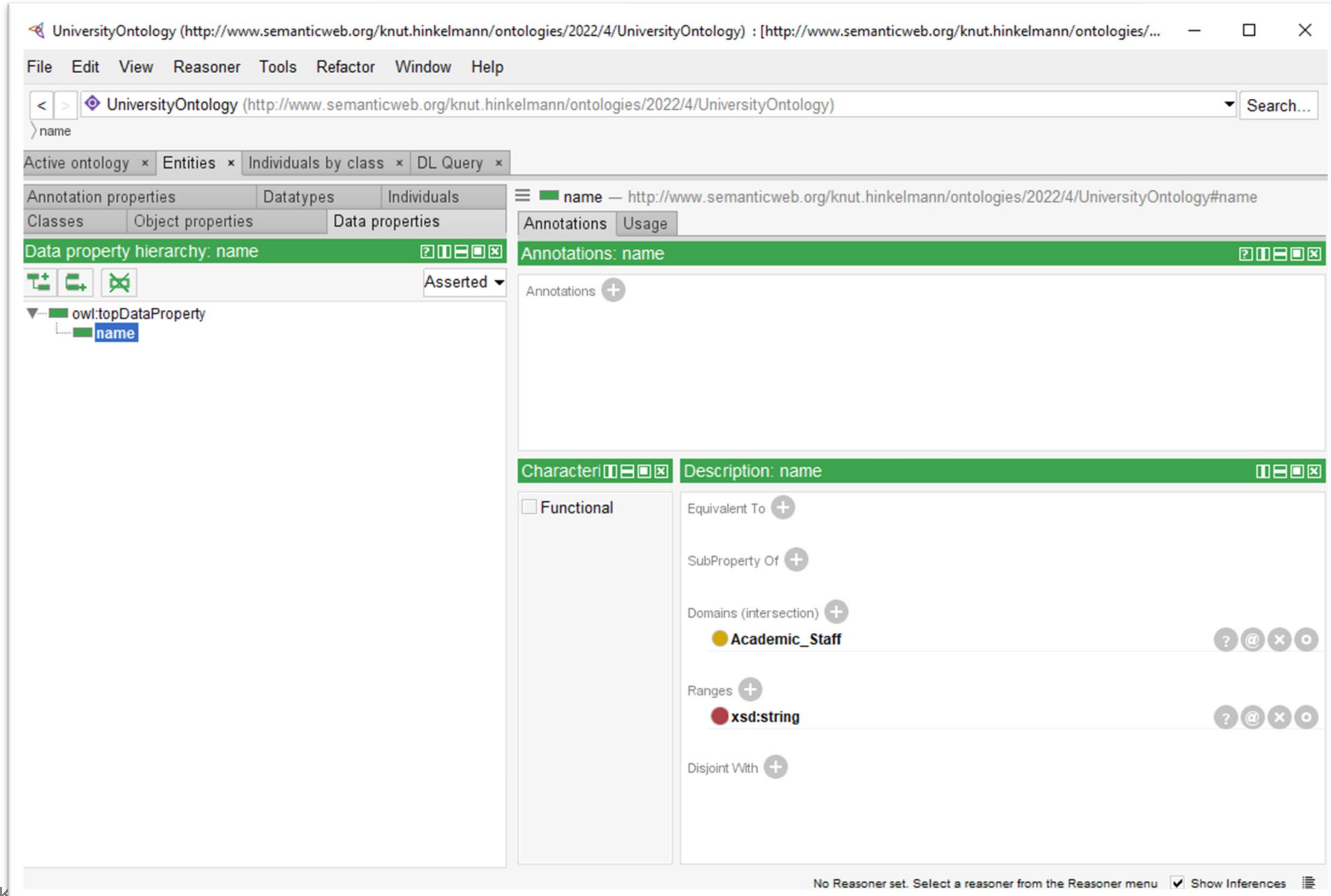
Object Property

```
:is_taught_by rdf:type owl:DatatypeProperty ;
              rdfs:subPropertyOf owl:topObjectProperty ;
              rdfs:domain :module;
              rdfs:range :lecturer .
```



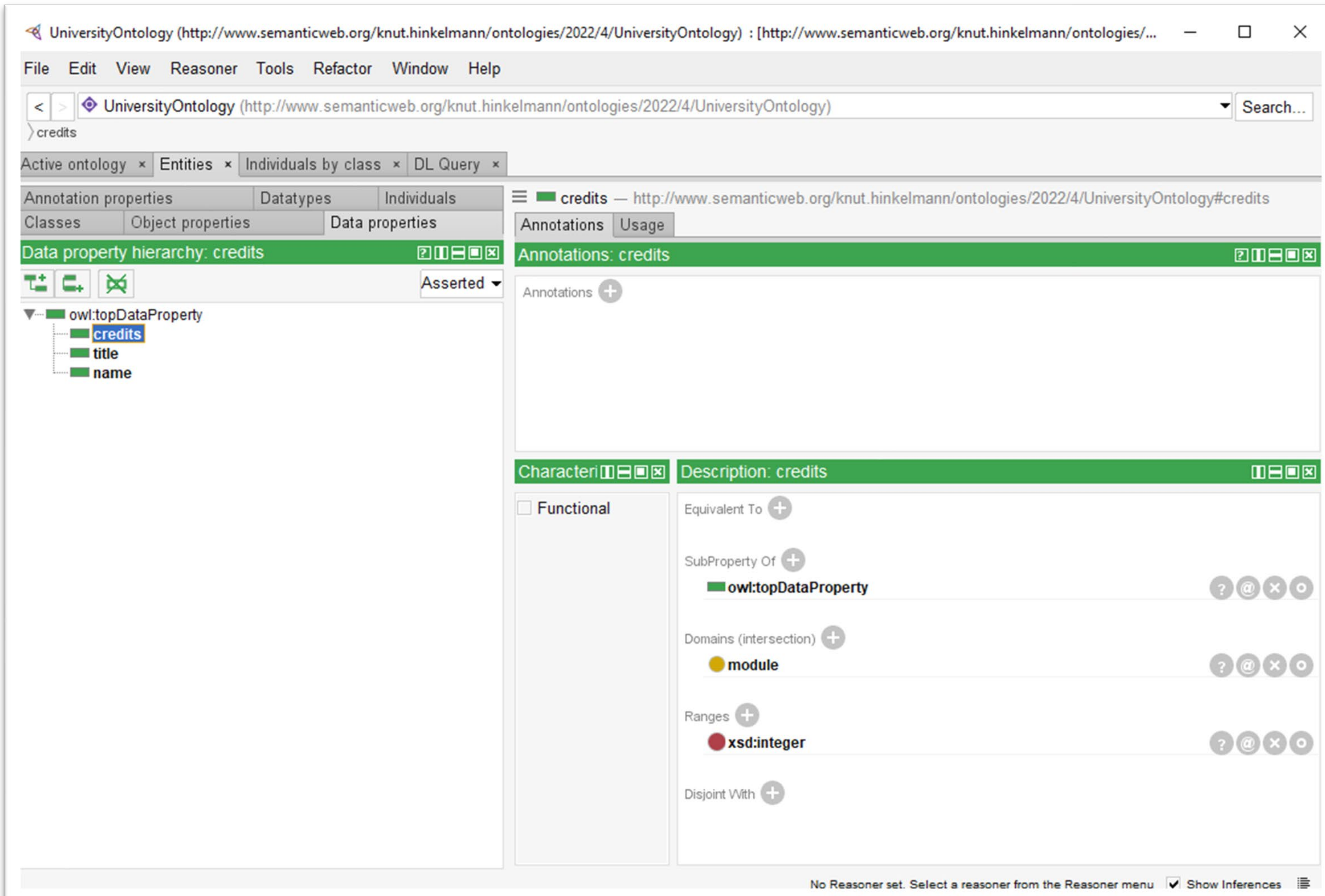
Data Property

```
:name rdf:type owl:DatatypeProperty ;
      rdfs:subPropertyOf owl:topDataProperty ;
      rdfs:domain :Academic_Staff ;
      rdfs:range xsd:string .
```

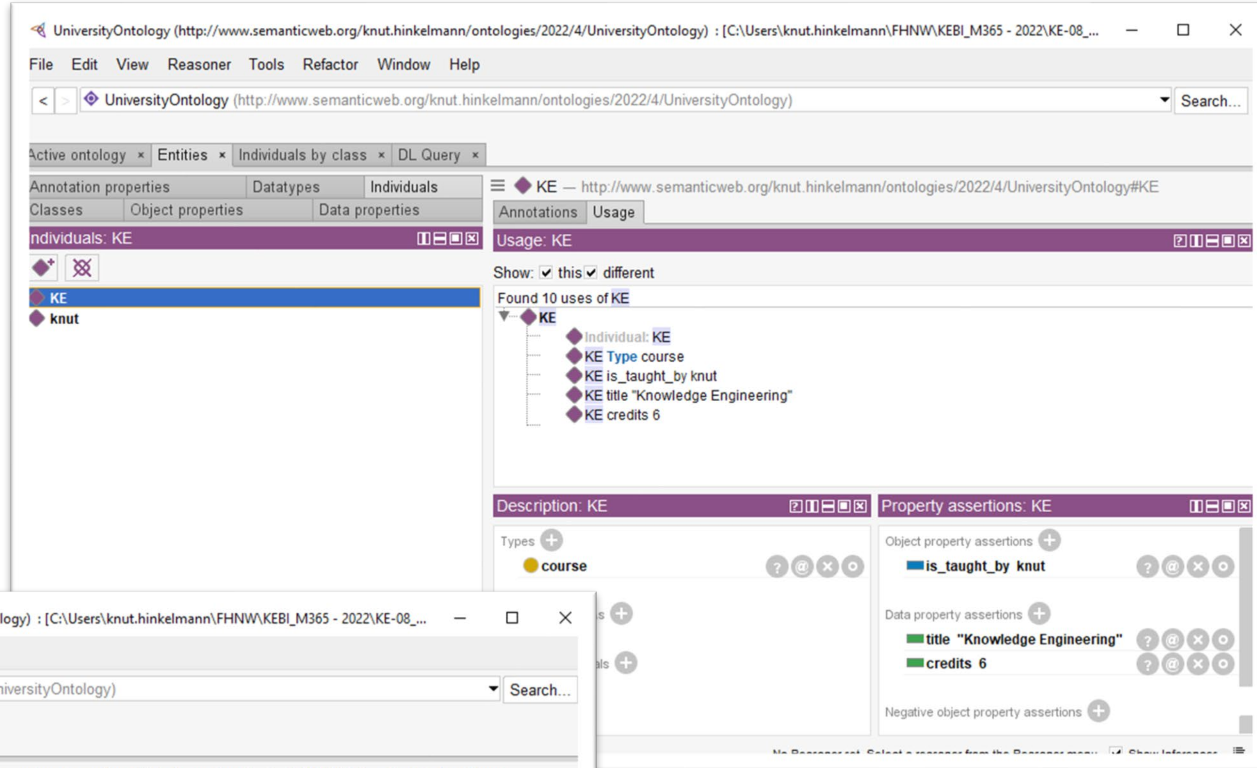


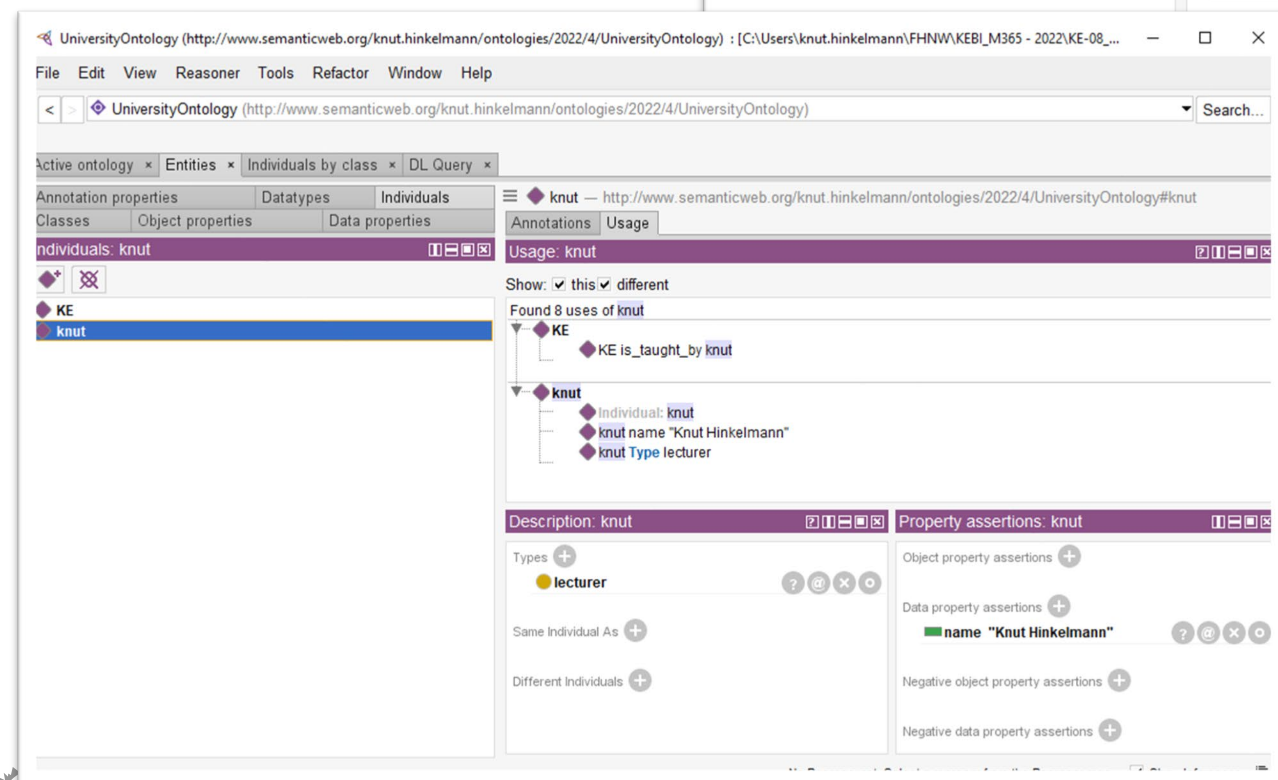
Data Property

```
:credits rdf:type owl:DatatypeProperty ;
         rdfs:subPropertyOf owl:topDataProperty ;
         rdfs:domain :module;
         rdfs:range xsd:integer .
```



Individuals



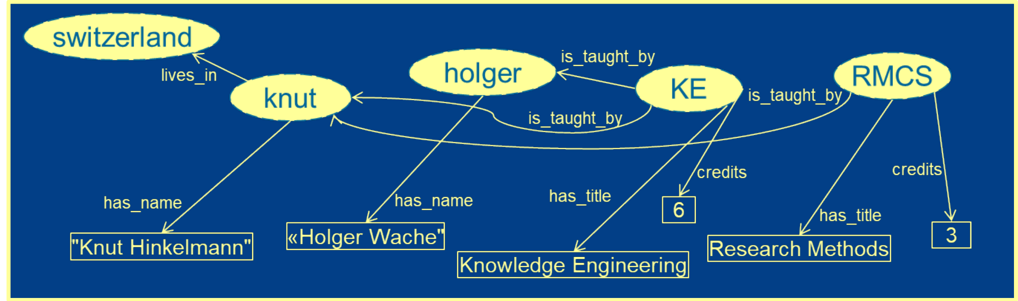



```
:KE rdf:type owl:NamedIndividual ,
             :course ;
    :is_taught_by :knut ;
    :credits 6 ;
     :title "Knowledge Engineering" .
:knut rdf:type owl:NamedIndividual ,
               :lecturer ;
      :name "Knut Hinkelmann" .
```

Exercise

- Add new class: country
- Add a property: A lecturer lives in a country
- Add new instance: Knut lives in Switzerland
- Add new classes and properties for the following knowledge
    - A project is a module
    - A Master Thesis is a project
    - Supervisor is a lecturer
    - A project has a supervisor
    - A project is performed by a student
- Add new instances
    - Giordano is a student who is performing a master thesis that is supervised by knut

Querying an Ontology



- Queries are mostly about navigating the graph in search of some patterns
- Sample types of queries
  - Navigating along a graph path, e.g who are the lecturers of KE

    SELECT ?x WHERE {:KE :is_taught_by ?x}
  - Navigating along a graph path with intermediate values, e.g. what are the names of the lecturers of KE

    SELECT ?x ?y WHERE {:KE :is_taught_by ?x.
    ?x :has_name ?y}
  - Navigating a path in reverse, e.g which modules is knut teaching

    SELECT ?x WHERE {?x :is_taught_by :knut}
  - Discover relationships, what is the relationship between KE and knut

    SELECT ?rel WHERE {:KE ?rel :knut}
  - Chain of relationships, what chain exists between KE and switzerland

    SELECT ?rel1 ?y ?rel2 WHERE {:KE ?rel1 ?y.
    ?y ?rel2 :switzerland}

Querying

SPARQL query:

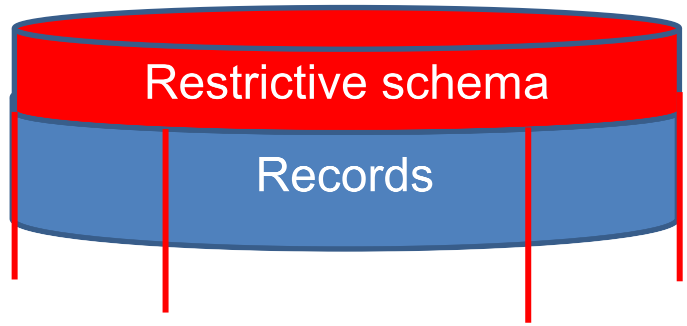
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?instance
        WHERE { ?instance rdf:type lecturer}
```

- Query Language: SPARQL
  - Variables: ?x
- Elements are denoted as URI
  - Prefixes for Abbrevations
    - Example: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- Sample query: Select all lecturers:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX uo: <http://www.semanticweb.org/knut.hinkelmann/ontologies/2020/4/UniversityOntology#>
SELECT ?subject
         WHERE { ?subject rdf:type uo:lecturer }
```

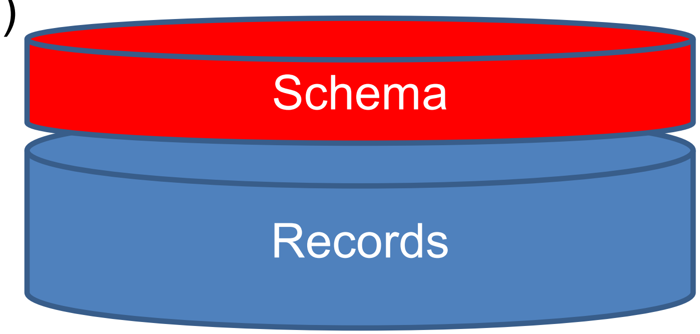
RDF Graphs vs Databases

In SQL databases, you cannot do anything before having a schema (the "DB structure")



In RDF graphs, ***schema is decoupled from "records"***

- Schema can be created after data
- Schema is optional (data can be queried in the absence of a schema)



Ontologies and Rules

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

SWRL – Semantic Web Rule Language

- SWRL is a rule language for the Semantic Web
- Rules are of the form of an implication between
  - an antecedent (body, condition) and
  - a consequent (head, conclusion)
- There are different representations for SWRL rules:
  - Human Readable Syntax
  - XML Concrete Syntax
  - RDF Concrete Syntax

https://www.w3.org/Submission/SWRL/

Human Readable Syntax

- Variables are same as SPARQL, indidated by ?

  ?x, ?something, ?object

- In the human readable syntAntecedent and consequent are separated by ⇒
  (in Protégé type -> instead of ⇒)

  hasParent(?x1,?x2) ^ hasBrother(?x2,?x3) -> hasUncle(?x1,?x3)

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

XML Concrete Syntax

- This is the XML Syntax of the uncle rule:

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#example1"/>
 <ruleml:_body>
   <swrlx:individualPropertyAtom  swrlx:property="hasParent">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x2</ruleml:var>
   </swrlx:individualPropertyAtom>
   <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
     <ruleml:var>x2</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_body>
 <ruleml:_head>
   <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
     <ruleml:var>x1</ruleml:var>
     <ruleml:var>x3</ruleml:var>
   </swrlx:individualPropertyAtom>
 </ruleml:_head>
</ruleml:imp>
```

Rules in Protege

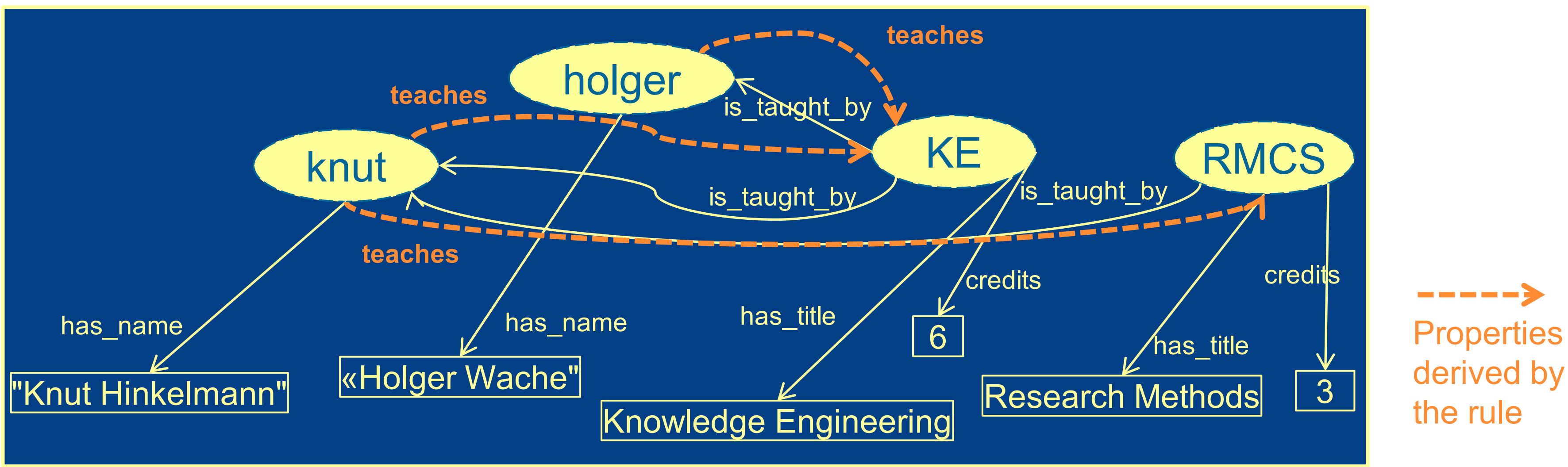
- The following rules derives the inverse of the proprerty is_taught_by

  ke:module(?l) ^ ke:is_taught_by(?c, ?l) -> ke:teaches(?l, ?c)

  - The rules means:

    If a course ?c is taught by lecturer ?l, then lecturer ?l teaches course ?c

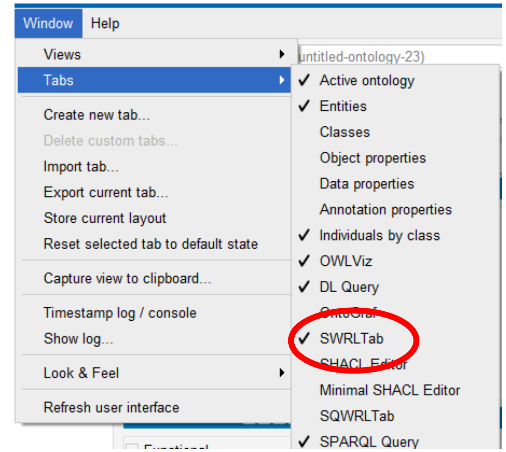
  - To run the rules there must be defined object property teaches has domain lecturer and range course

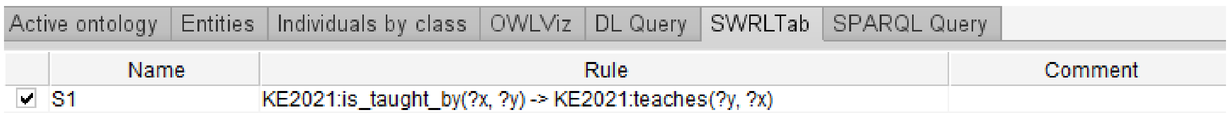


Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

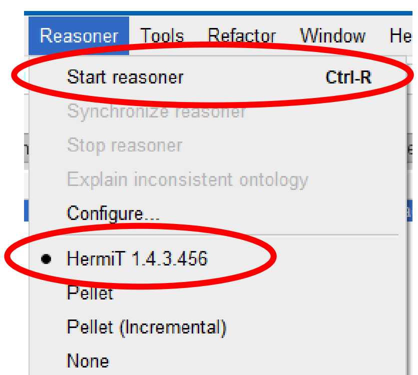
Rules in Protege

- In Protege there is a SWRLTab
- In this tab you specify rules





- To execute the rules, a reasoner must be started
  - In the menu Reasoner select reasoner HermiT and start the reasoner

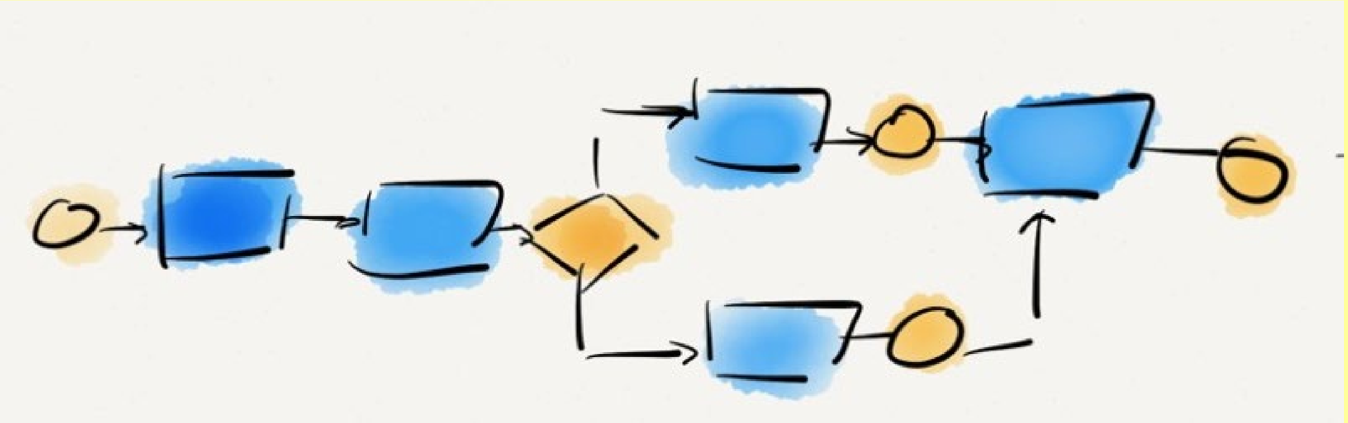


Ontology Development 101

(Noy & McGuinness 2001)

Exercise: Modeling Process Knowledge in an Ontology

- We create a knowledge base for process knowledge
  - ♦ Define the ontology
  - ♦ Represent knowledge of a process



Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Ontology Development 101

- Determine the domain and scope of the ontology
- Consider reusing existing ontologies
- Enumerate important terms
- Define classes and class hierarchy
- Define the data and object properties of classes
- Define the facets of properties
- Create instances

7

Determine the domain and scope of the ontology

- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers? → Competency questions
- Who will use and maintain the ontology?

Competency Questions

- One of the ways to determine the scope of the ontology is to sketch a list of questions that a knowledge base based on the ontology should be able to answer (Gruninger and Fox 1995)
  - Does the ontology contain enough information to answer these types of questions?
  - Do the answers require a particular level of detail or representation of a particular area?

- Exercise: We want to represent knowledge about
  - the process flow
  - Responsibilies for tasks
- Competency Questions:
  - Who executes task X?
  - Which task is executed after task X?
  - When can task X start?
- Sample process:

The waiter serves the beverages. Then the waiter serves the food. When the guests are finished, the waiter presents the bill.

Consider reusing existing ontologies

- It is always worth considering what others have done, and check if their work can be refined and extended for our particular domain and task
- Mandatory if the system needs to interact with other applications that have already committed to particular ontologies or controlled vocabularies

- Are there already ontologies for business processes?
- What source can we use to create an ontology for business processes?

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Enumerate important terms in the ontology

- What are the terms we would like to talk about?
- What are their properties?
- What would we like to say about those terms?

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

- Taking into account your knowledge about business processes modelling, what are important terms that are needed to answer the competency questions?
  - Who executes task X?
  - Which task is executed after task X?
  - When can task X start?

Define Classes and Class Hierarchy

- Several possible approaches in developing a class hierarchy:
  - Top-down: General to specific concepts
  - Bottom-up: Specific to general concepts
  - Combination: Salient to general and specific concepts
- Classes for
  - Modeling Objects
  - Relations

- Taking into account your knowledge about business processes modelling, how can we create a class hierarchy?
- Which terms should be classes?
- What are subclasses?

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Define the properties of classes

- Describe the internal structure of concepts
  - Data Properties: Attributes
    - Range are data typles like String, Integer, …
  - Object Properties: Relations to other concepts
    - Range are Classes
- Inheritance to Subclasses

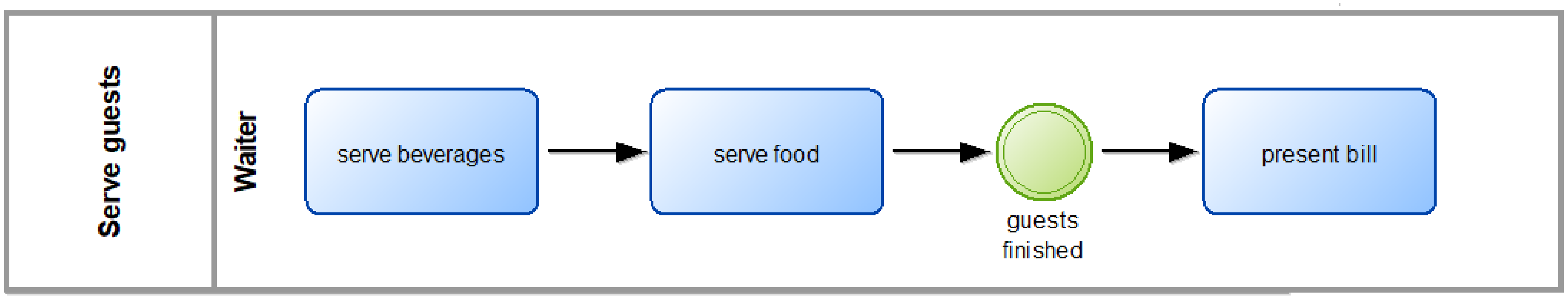
- Which data and object properties make sense for modelling business processes?

Create Instances

- Model a business process in an ontology

The waiter serves the beverages. Then the waiter serves the food. When the guests are finished, the waiter presents the bill.

Modeling Business Processes as graphical models is more adequate



Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

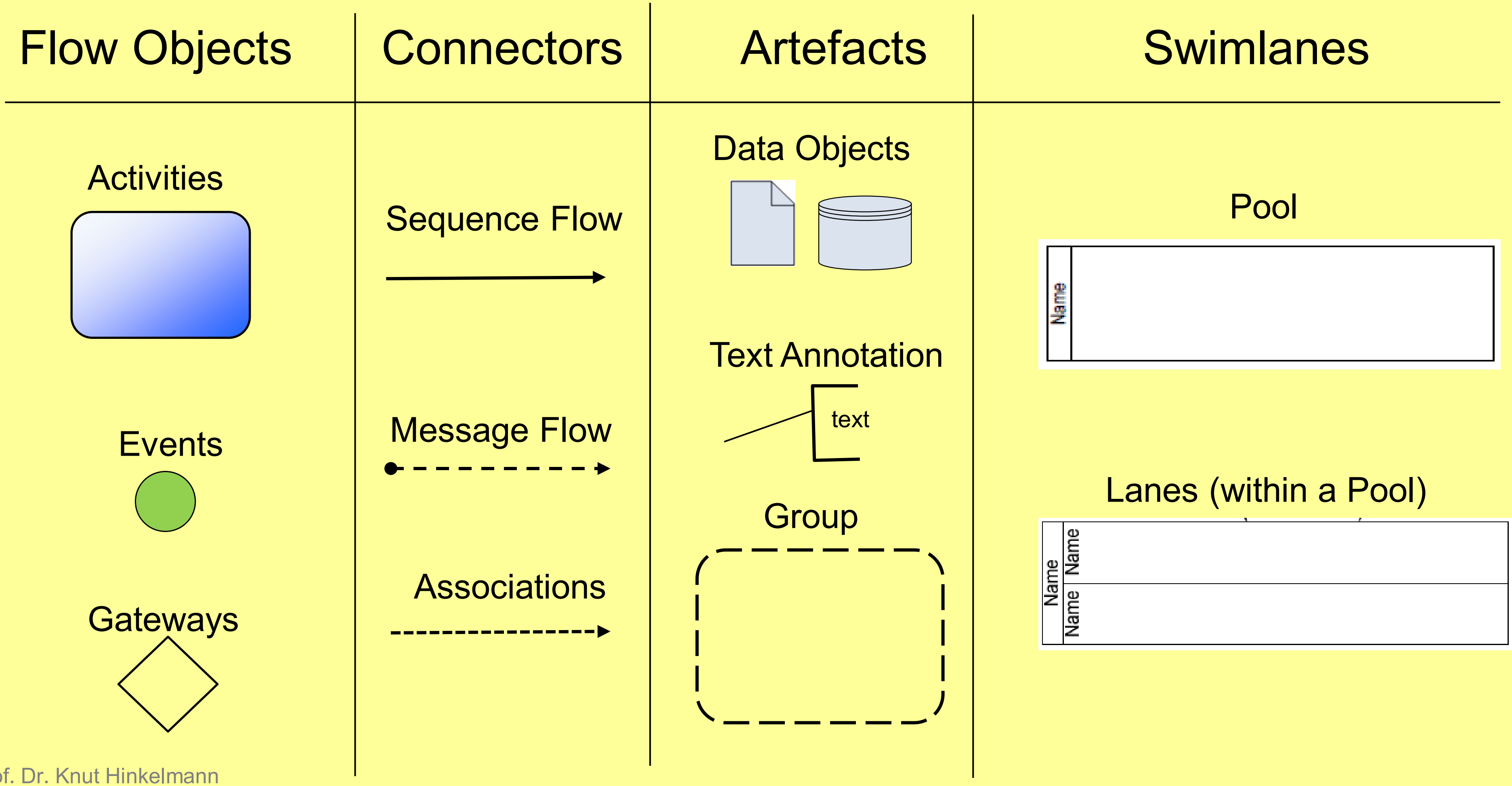
Queries

- Write queries for the following questions
  - Who performs task «Serve food»
  - When can task «Present Bill» start

Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch

Elements of BPMN

Elements of BPMN can be divided into 4 categories:



Prof. Dr. Knut Hinkelmann
knut.hinkelmann@fhnw.ch